

**Mini Project Report**

*on*

**BODY METER**

*Submitted by*

**AKHILA HARIDAS  
ASWATHY A G  
SABITHA SUBAIR**

*In partial fulfillment for the award of the degree of*

**B. TECH DEGREE**

*in*

**COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF ENGINEERING**

**COCHIN UNIVERSITY OF SCIENCE &  
TECHNOLOGY  
KOCHI-682022**

**MARCH 2014**

**Division of Computer Engineering School of  
Engineering  
Cochin University of Science & Technology  
Kochi-682022**

---

**CERTIFICATE**

*Certified that this is a bonafied record of the project work titled*

***Body Meter***

*Done by*

***Akhila Haridas  
Aswathy A G  
Sabitha Subair***

*of VI semester Computer Science & Engineering in the year 2014 in partial fulfillment  
of the requirements for the award of Degree of Bachelor of Technology in Computer  
Science & Engineering of Cochin University of Science & Technology.*

***Preetha S***  
*Project Guide*

***Dr. Pramod Pavithran***  
*Head of the Division*

## **ACKNOWLEDGEMENT**

We extend our sincere and heartfelt thanks to our esteemed **guide, Mrs.Preetha.S** and for her exemplary guidance, monitoring and constant encouragement throughout the course at crucial junctures and for showing us the right way.

We would like to extend thanks to our respected **Head of the division, Dr. Pramod Pavithran** for allowing us to use the facilities available. We would like to thank other faculty members also .Last but not the least,

We would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

**AKHILA HARIDAS**

**ASWATHY A G**

**SABITHA SUBAIR**

## **CONTENTS**

### **1. ABSTRACT**

### **2. INTRODUCTION**

### **3. SYSTEM STUDY**

3.1. Proposed system

3.2. Overview of software

3.3. Hardware specification

3.4. Software requirements

### **4. SYSTEM DESIGN**

4.1. Dataflow diagram

### **5. SYSTEM IMPLEMENTATION**

5.1. Testing

### **6. DATABASE SCHEMA**

### **7. CONCLUSION**

### **LIST OF FIGURES**

### **LIST OF ABBREVIATIONS**

### **APPENDIX-A**

### **APPENDIX-B**

### **REFERENCES**

## **ABSTRACT**

During the busy life, people don't find time to keep track of their health and may lead to critical medical conditions. A solution is to record all readings related to health and later verify it manually. But with help of a computer software we can store the readings digitally and these data can be processed for productivity.

Record your personal test results as in a diary. Weight, Diabetes, BP, Height etc with date. The data recorded can be viewed in graphs and charts. These data is used for calculating an expected result in future by analyzing current status using normal distribution equations and interpolation formulas. The software also reminds the future test dates and it can produce a warning if the readings are growing worse. The software uses a DB, and it enables multiple user profiles, hence individuals can use the software. This software can also be used by Doctors to keep track of their patients.

## **1. INTRODUCTION**

During the busy life, people don't find time to keep track of their health and may lead to critical medical conditions. A solution is to record all readings related to health and later verify it manually. But with help of a computer software we can store the readings digitally and these data can be processed for productivity.

Record your personal test results as in a diary. Weight, Diabetes, BP, Height etc with date. The data recorded can be viewed in graphs and charts. These data is used for calculating an expected result in future by analyzing current status using normal distribution equations and interpolation formulas. The software also reminds the future test dates and it can produce a warning if the readings are growing worse. The software uses a DB, and it enables multiple user profiles, hence individuals can use the software. This software can also be used by Doctors to keep track of their patients.

### **1.1. PURPOSE**

This software can store the readings digitally and these data can be processed for productivity and for calculating an expected result in future by analyzing current status.

## **1.2. SYSTEM SCOPE**

### **Scope of the software:-**

- Reminds patients about their tests.
- Shows warnings and predictions.
- Helps to avoid critical medical conditions.
- Data stored is secured.
- Doctors can keep track of their patients.
- Keeps track of all dates and appointments.

## **2. SYSTEM ANALYSIS**

This chapter deals with the analysis of the system proposed by our team. It covers the features, advantages and disadvantages of both the system.

### **2.11 Advantage**

- No loss of document due to power loss or database errors.
- No requirement computers knowledge
- No need of costly hardware used in the system.

### **2.12 Disadvantage**

- Difficult to maintain all entries.
- Difficult to generate a report.
- It is very much time consuming.
- Difficult to search a record.
- Data may not be perfect human error.

It contains various forms and reports with different function. It has the menus created for different forms and reports. When you click the menu, it opens the related forms about the saving account, login form, inquiry, etc.

## **2.2. PROPOSED SYSTEM**

The proposed system is having many advantages over the existing system. It requires much less overhead and very efficient.

### **2.2.1 Economical feasibility**

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a



candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system.

### **2.2.2 Technical feasibility**

The technical feasibility in the proposed system deals with the technology used in the system. It deals with the hardware and software used in the system whether they are of latest technology or not. It happens that after a system is prepared a new technology arises and the user wants the system based on that technology. Thus it is important to check the system to be technically feasible.

### **2.2.3 Operational feasibility**

The project has been developed in such a way that it becomes very easy even for a layman with little computer knowledge to operate it. The software is very user friendly and does not require any technical person to operate. Thus the project is even operationally feasible.

## **2.3. OVERVIEW OF SOFTWARE**

### **JSP Overview**

Java Server Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with `<%` and end with `%>`.

A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

### **Why Use JSP?**

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
- JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of J2EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

### **Advantages of JSP:**

Following is the list of other advantages of using JSP over other technologies:

- ☐ **Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- ☐ **Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of `println` statements that generate the HTML.
- ☐ **Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.
- ☐ **JavaScript:** JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.
- ☐ **Static HTML:** Regular HTML, of course, cannot contain dynamic information.

### **JSP Compilation:**

When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.

The compilation process involves three steps:

- ☐ Parsing the JSP.

- ☐ Turning the JSP into a servlet.
- ☐ Compiling the servlet.

### **JSP Initialization:**

When a container loads a JSP it invokes the `jspInit()` method before servicing any requests. If you need to perform JSP-specific initialization, override the `jspInit()` method:

```
public void jspInit(){ // Initialization code... }
```

Typically initialization is performed only once and as with the servlet `init` method, you generally initialize database connections, open files, and create lookup tables in the `jspInit` method.

### **JSP Execution:**

This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.

## **2.4. HARDWARE SPECIFICATION**

- Processor - Pentium iv and above
- Ram - 256 MB and above
- Hard disk - 1GB or above

## **2.5. SOFTWARE REQUIREMENT**

- Programming Language - Java
- Operating System - Windows 8
- Web Technology - JSP
- Database - My Sql

### *Body Meter*

---

- Web Server - Apache Tomcat 7.0
- Environment - IntelliJ IDEA

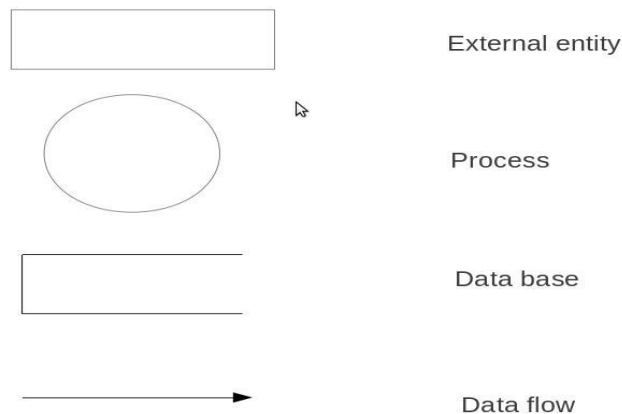
### 3. SYSTEM DESIGN

The system design is needed for information processing technology and the user interface development analysis. It contain a high level overview of the organization in which the common activities, processes and products are described in relation to how they create, use and modify information.

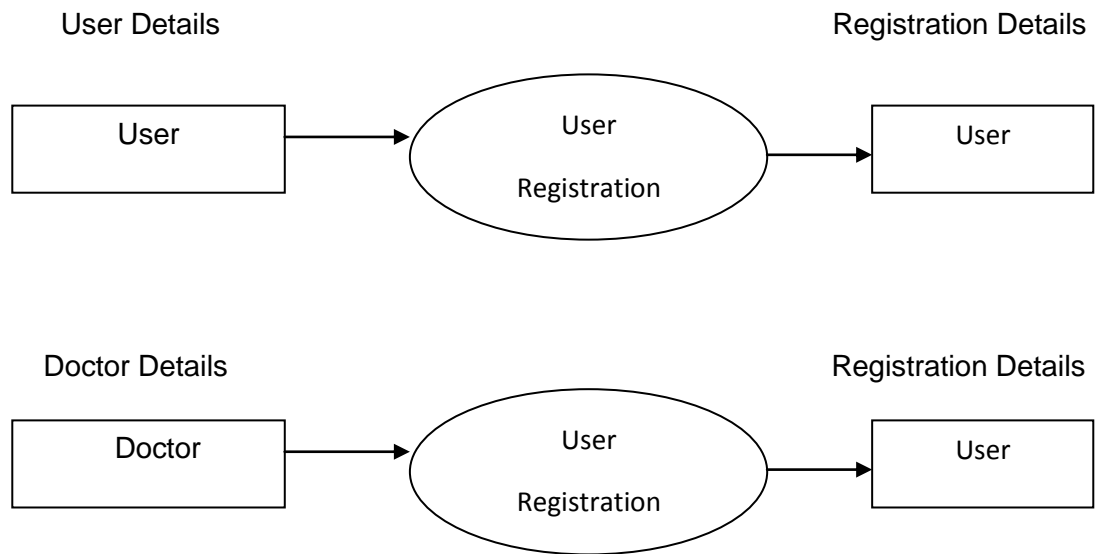
Below given dfd's can be used to describe the overall user interface:

#### Data Flow Diagram

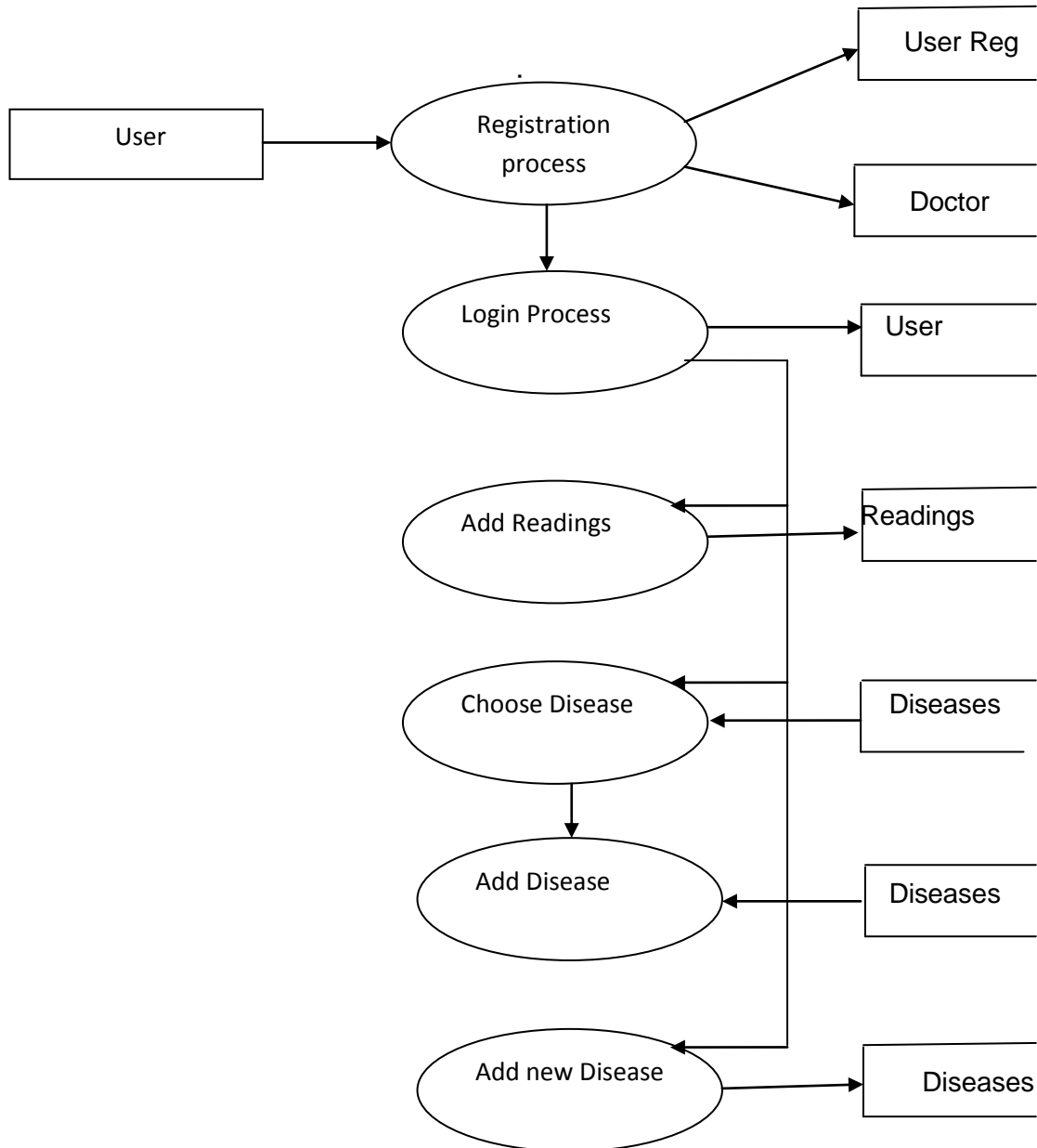
##### Notation Used



**LEVEL- 0 DFD:**

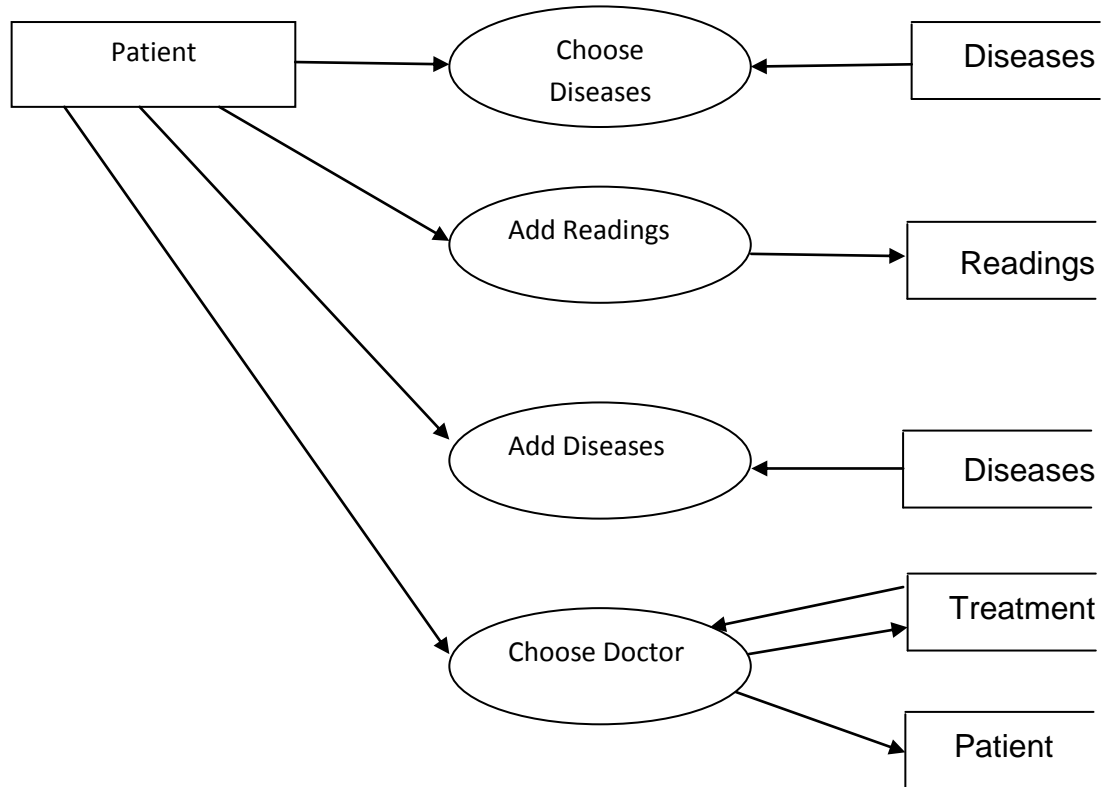


**LEVEL-1.1 DFD:**

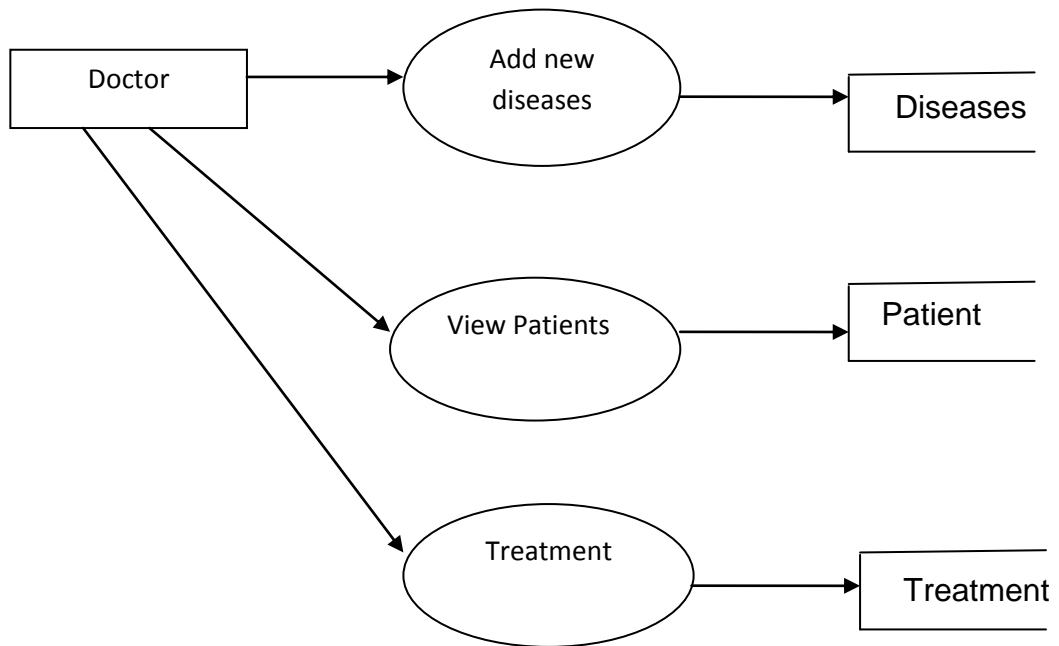


**LEVEL-1.2 DFD:**





**LEVEL-1.3 DFD:**



## **5. SYSTEM TESTING**

System testing is normally carried out in a planned manner according to the system test plan document. The system test plan identifies all testing-related activities that must be performed, specifies the schedule of testing, and allocates resources. It also lists all the test cases and the expected outputs for each test case. Here the modules are integrated in a planned manner.

### **5.1. FUNCTIONAL TESTING**

Functional testing refers to tests that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work". Some examples of functional testing done in our project:

1. By checking all the login modules, it is ensured that only registered users can access all the facilities.
2. All the agents are verified by their IEC(Import Export Code)

### **5.2. STRUCTURAL TESTING**

Structural testing is also called White box testing. This means a testing technique whereby explicit knowledge of the internal workings of the item being tested is used to select the test data. White box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its intended goal. White box testing does not account for

errors caused by omission, and all visible code must also be readable.

### **5.3. SYSTEM TESTING**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limiting type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing is an investigatory testing phase, where the focus is to have almost a destructive attitude and tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

### **TEST CASES**

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not.

### **Unit Test Cases**

The software is being divided into different components and unit testing is performed on each of these modules. This section is repeated for all components.

### **Integration Test cases**

Integration testing is a part of stress testing which involves integrating the components to create a system or sub-system. It may involve testing an increment to be delivered to the customer. In integration testing,

the test team has access to the system source code. The system is tested as components are integrated.

### **Validation Test cases**

This testing is done to see whether the integrated software is valid according to the user needs.

## **6. DATABASE SCHEMA**

The goal of database design is to ensure that the data is represented in such a way that there is no redundancy and no extraneous data is generated, thus generate relationship in as high an order as possible. Having identified all the data on the system it is necessary to at the logical database design. Database design involves designing the conceptual model of the database. This model is independent of the physical representation of the data. Once the conceptual model is designed, it can be mapped to the DBMS/RDBMS that is actually being used.

Following are tables that have been used in this project:-

### **1. TABLES: USER DETAIL**

<b>COLUMN NAME</b>	<b>DATA TYPE</b>	<b>CONSTRAINTS</b>
User Id	Int	Primary Key
Password	Int	
First Name	Varchar(20)	
Last Name	Varchar(20)	
Address	Varchar(20)	
Occupation	Varchar(50)	
Email Id	Varchar(20)	
Dob	Int	
Type	Varchar(20)	
Contact No:	Long Int	

**2. TABLE: PATIENT**

COLUMN NAME	DATA TYPE	CONSTRAINT
Pat_Id	Int	Primary Key
User_Id	Int	
Doc_Id	Int	
Dis_Id	Int	

**3. TABLE:TREATMENT**

COLUMN NAME	DATA TYPE	CONSTRAINT
Treat-Id	Int	Primary Key
Dis_Id	Int	
Name	Varchar(20)	

**4. TABLE: DOCTOR**

COLUMN NAME	DATA TYPE	CONSTRAINT
Doc_Id	Int	Primary Key
User_Id	Int	
Specialized_In	Varchar(20)	

**5. TABLE: READINGS**

COLUMN NAME	DATA TYPE	CONSTRAINT
Reg_Id	Int	Primary Key
Pat_Id	Int	
User_Id	Int	
Name	Varchar(30)	

**6. TABLE:DISEASES**

COLUMN NAME	DATA TYPE	CONSTRAINT
Dis_Id	Int	
Name	Varchar(30)	
Symptoms	Long Text	
Treatment_Id	Int	



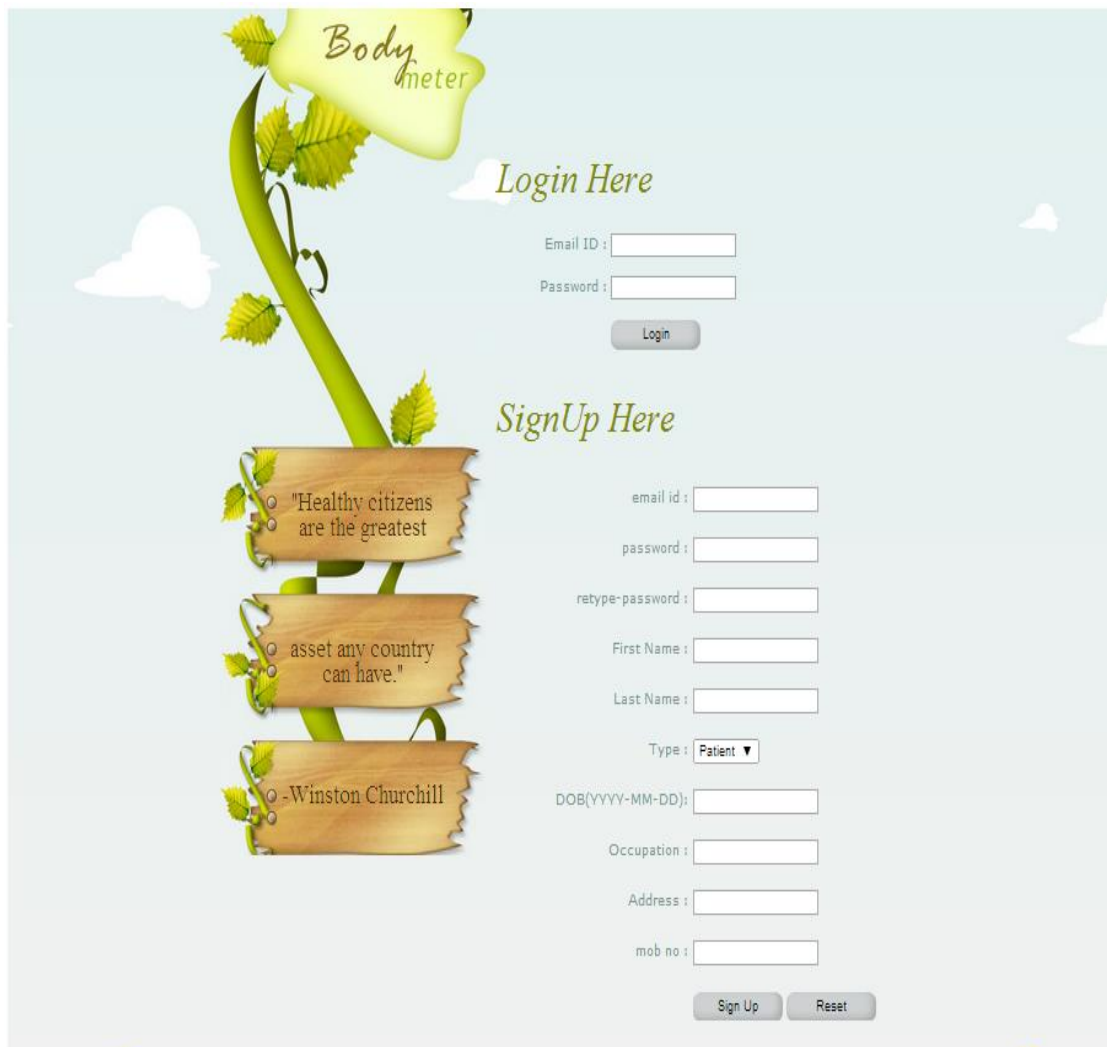
## **7. CONCLUSION**

The software is designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project:-

- Automation of the tire system improves the efficiency.
- It provides a friendly graphical user interface.
- This application will avoid the manual work.

It records all readings related to health and later verify it, these data can be processed for productivity. These data is used for calculating an expected result in future by analyzing current status using normal distribution equations and interpolation formulas. The software also reminds the future test dates and it can produce a warning if the readings are growing worse.

## LIST OF FIGURES



The image shows a web interface for 'Body Meter'. On the left, a green vine with yellow leaves and three wooden signs is positioned. The signs contain the following text: "Healthy citizens are the greatest asset any country can have." and "-Winston Churchill". The top sign is yellow and says "Body meter". To the right of the vine, there are two main sections: "Login Here" and "SignUp Here". The "Login Here" section has fields for "Email ID" and "Password", and a "Login" button. The "SignUp Here" section has fields for "email id", "password", "retype-password", "First Name", "Last Name", "Type" (a dropdown menu with "Patient" selected), "DOB(YYYY-MM-DD)", "Occupation", "Address", and "mob no". There are "Sign Up" and "Reset" buttons at the bottom of the registration section.

**Body meter**

*Login Here*

Email ID :

Password :

Login

*SignUp Here*

email id :

password :

retype-password :

First Name :

Last Name :

Type : Patient ▼

DOB(YYYY-MM-DD):

Occupation :

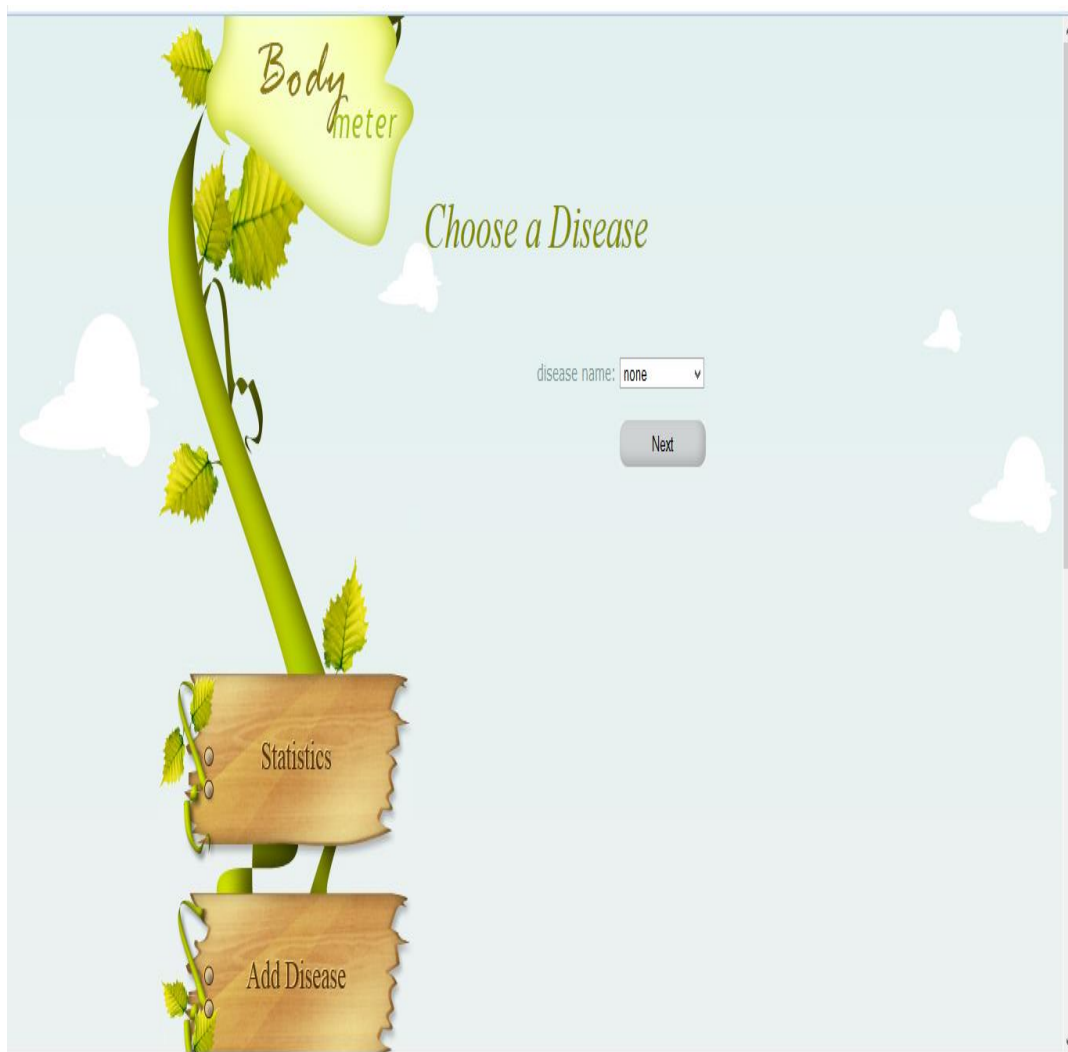
Address :

mob no :

Sign Up Reset

"Healthy citizens are the greatest asset any country can have."

-Winston Churchill



The image shows a web application interface for 'Body Meter'. On the left, there is a decorative graphic of a green vine with yellow leaves. The main header area has a light blue background with a large yellow leaf-like shape containing the text 'Body meter'. To the right of this, there is a navigation menu with links: 'Home', 'About us', 'Library', 'News', 'Blog', and 'Register'. Below the navigation menu, the title 'Add Readings of Diabetes mellitus' is displayed in a large, elegant, dark green font. The form area contains three input fields: a date field with the value '2014/03/12', a text field labeled 'BG fasting under 100' with the value '120', and another text field labeled 'BG fasting 100-125' with the value '150'. Below these fields is a grey 'save' button.

Body meter

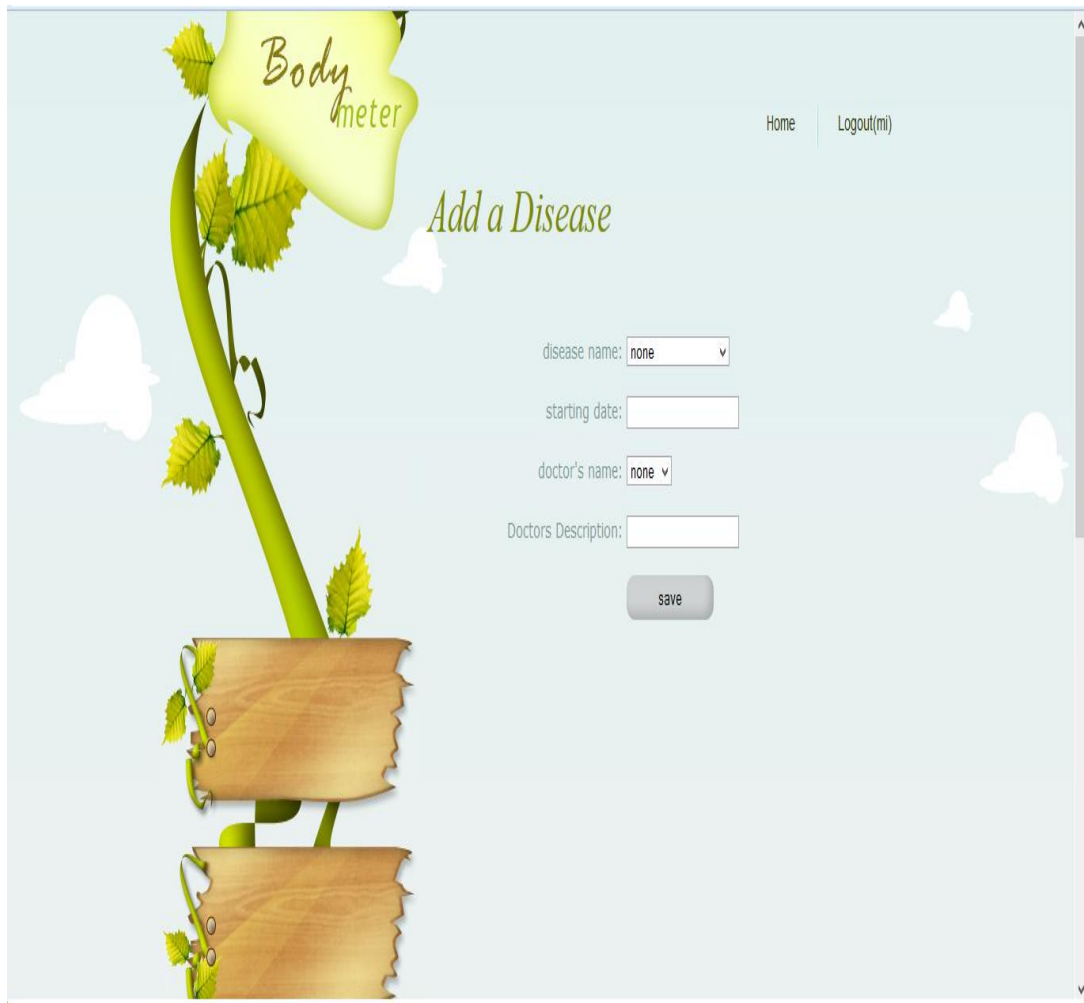
[Home](#) [About us](#) [Library](#) [News](#) [Blog](#) [Register](#)

## Add Readings of Diabetes mellitus

date:

BG fasting under 100

BG fasting 100-125



The image shows a web application interface for 'Body Meter'. On the left, there is a decorative graphic of a green vine with yellow leaves growing out of two wooden crates. A yellow leaf at the top of the vine contains the text 'Body meter'. The background is a light blue sky with white clouds. In the top right corner, there are links for 'Home' and 'Logout(mi)'. The main heading is 'Add a Disease'. Below this, there are four input fields: 'disease name:' with a dropdown menu showing 'none', 'starting date:' with a text box, 'doctor's name:' with a dropdown menu showing 'none', and 'Doctors Description:' with a text box. A 'save' button is located below the description field.

Body meter

Home Logout(mi)

## Add a Disease

disease name: none ▾

starting date:

doctor's name: none ▾

Doctors Description:

save

Body meter

Home Logout(doc)

## Add a New Disease

Disease name:

Disease Symptoms:

Disease Description:

Treatments:

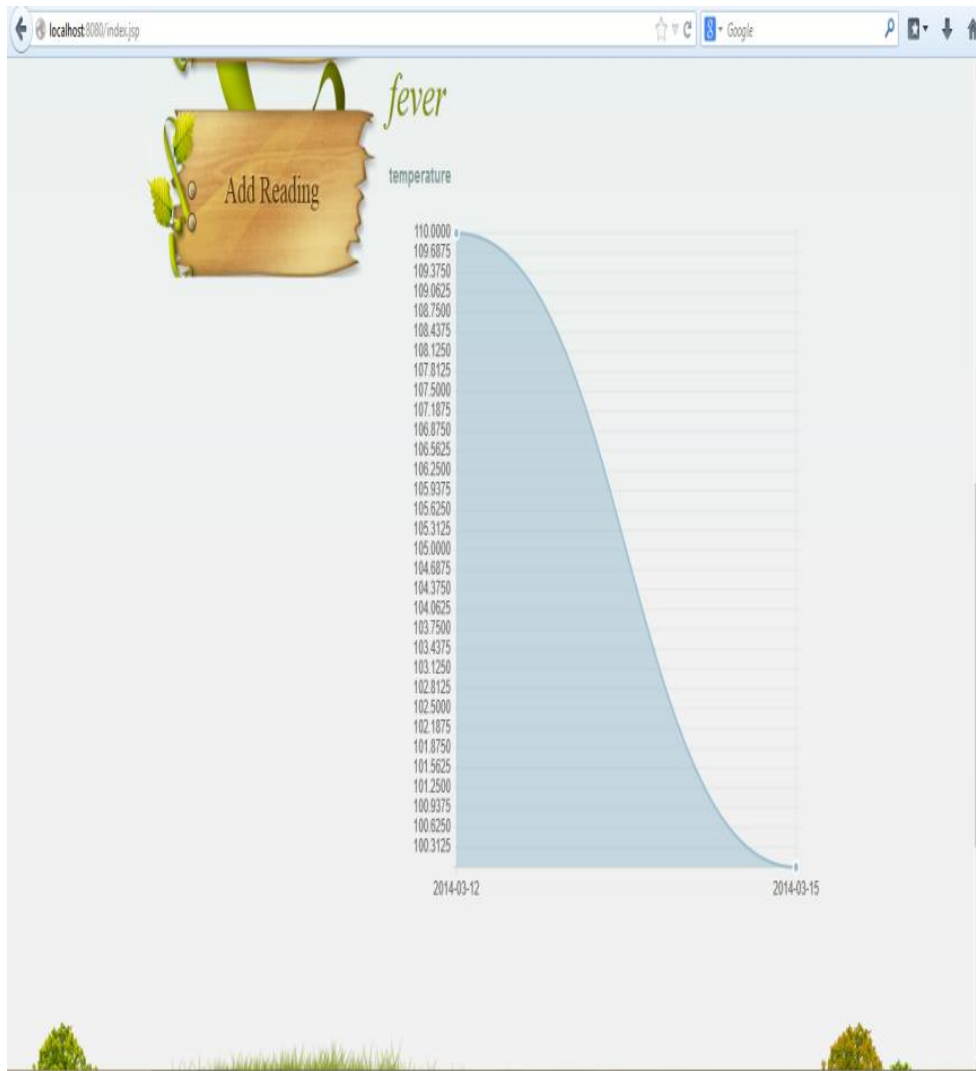
Treatment Name
Safe Min
Safe Max

Statistics

Add New Disease

## Body Meter

---



## **ABBREVIATIONS**

SQL	-	Structured Query Language
JDK	-	Java Development Kit
JDBC	-	Java Database Connectivity
JSP	-	Java Server Page
CSS	-	Cascading Style Sheets
HTML	-	Hyper Text Markup Language
J2EE	-	Java Enterprise Edition



## **APPENDIX – A**

### **SYSTEM SECURITY:**

#### **Password Encryption: UTF-8**

#### **Password Modification:**

Admin can only change the password.

Code corresponding to above is given below:

#### **SpasswordChangeAction.jsp**

```
<%
{
    String uname=request.getParameter("una");
    String pass=request.getParameter("upass");
    if(cb.updatePass(uname,pass))
    {
        response.sendRedirect("SPassword.jsp?v=1");
    }
Else
{
    response.sendRedirect("SPassword.jsp?v=2");
}
%>
```

## **APPENDIX – B**

### **CONNECTION ESTABLISHMENT**

```
public ConnectiondbConnection()
{
    try
    {
        Class.forName("com.mysql.jdbc.Driver");

        conn = DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/dbcuess22", "root","mysql");

    }

    catch (SQLException)

    {
        e.printStackTrace();

    }

    catch (ClassNotFoundException e)
    {
        e.printStackTrace();

    }

    return conn;
```

## **REFERENCES**

1. The J2ee 1.4 Tutorial “Sun Java System Applications Platform Edition” By Eric Armstrong & Ian Evans.
2. The Complete Reference Java “Tata McGraw Hill Publication “By Herbert Schildt.
3. My sql Enterprise Solutions “Wiley Publication” B Alexander Sasha Pachev.
4. My sql And Java Developer’s Guide “Wiley Publications By Mark Matthews & Jim Cole.